TIE-02401 Ohjelmoinnin tekniikat Syksy 2017

Harjoitustyöohje: Galaxy Quest

Palaa etusivulle

Galaxy Quest

Syksyn 2017 harjoitustyönä toteutetaan intergalaktinen seikkailupeli, työnimeltään Galaxy Quest. Harjoitustyössä työskentelee neljän hengen ohjelmistotiimi. Sinä ja parisi muodostatte osan tiimiä, vastuuhenkilö ja harjoitustyöassistenttinne täydentää tiimiänne kurssin puolelta. Tiimin osien välillä vallitsee työnjako ja koodin yhdistämiseksi on sovittu yhteisestä rajapinnasta, jonka yli koodinne keskustelee kurssitiimin koodaaman puolen kanssa.

1. Yleistä

Avaruusaluksia liikkuu galaksissa tähtijärjestelmien välillä. Avaruudessa on avaraa, joten aluksia on paljon ja usein niille sattuu ja tapahtuu. Sinun tehtäväsi avaruuslaivasi kapteenina on kulkea tähtijärjestelmien välillä ja etsiä pulaan joutuneita aluksia. Pelin lopullisen tavoitteen voit päättää itse...

Pelin kuvaus

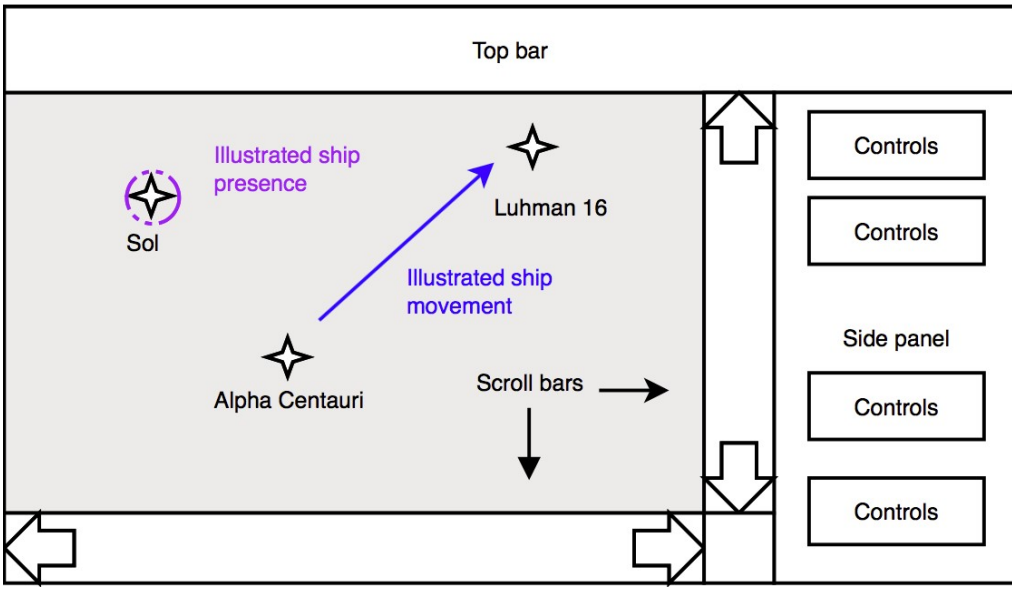
Peliä pelataan näkymällä, jossa on esitetty tähtijärjestelmiä ja liikkeellä olevia aluksia. Lisäksi pelissä kulkee pelattava alus. Aluksilla on erilaisia moottoreita. Ainakin seuraavilla moottoreilla varustettuja aluksia on liikkeellä:

- poimuajomoottori
- madonreikämoottori

Pelattava alus on varustettu tunnetulla moottorilla ja pelaaja liikkuu aluksella tähtijärjestelmästä toiseen.

Toteutus

Peliin on toteutettu valmiiksi pelin tilan hallinta, kuten ei-pelattavat alukset ja käytettävissä olevat perusmoottorityypit. Itse toteutettavaksi jäävät pelin graafinen käyttöliittymä, pelattavan aluksen toiminta ja esittäminen sekä pelin tavoite ja tilastointi. Alla on hahmotelma siitä, miltä pelin käyttöliittymä saattaisi näyttää.



Monet pelin yksityiskohdat on jätetty pelin kuvauksessa tarkoituksella avoimiksi ja ne voi toteuttaa haluamallaan tavalla. Tällaisia ovat mm.

- Pelin alkutilanne: missä pelattava alus sijaitsee?
- Pelin tavoite: pelastusoperaatio? avaruusrosvous? galaksien välinen sota? jokin ihan muu?
- Vuorojen kulku: mitä pelivuoron aikana tapahtuu? onko peli vuoropohjainen vai kuluuko aika tasaisesti?
- Pelin päättyminen: Miten pelin loppu ja siinä menestyminen määräytyvät?

Käyttötapaus

Seuraava toimintojen sarja havainnollistaa pelin käynnistystä ja toimintaa. Kyseessä on esimerkki. Oma pelinne voi toimia toisin kuin mitä tässä on kuvattu.

- Peliohjelma käynnistetään itse toteutetun pääohjelman avulla. Peliohjelma alustaa kurssin puolen toteuttamat tapahtumakäsittelijän ja pelilogiikan ajurin, luo itse toteutetun galaksiolion ja asettaa tähtijärjestelmät, ei-pelattavat alukset, pelaajan ym. alkutilanteen yksityiskohdat. Peliohjelma kysyy erillisellä konfiguraatio-ikkunalla ei-pelattavien alusten määrän ja muuta pelin tarvitsemaan alkutietoa. Tämän jälkeen se avaa itse toteutettavan pääikkunan, jossa näkyvät tähtijärjestelmät ja kaikki alukset.
- Pelaaja aloittaa pelin klikkaamalla nappulaa pääikkunassa.
- Pelaaja valitsee kohdetähtijärjestelmän ja aloittaa matkan sinne.
- Pelaaja saapuu kohdetähtijärjestelmään ja löytää sieltä pulaan joutuneen aluksen. Pelaaja korjaa aluksen ja kerää näin itselleen pisteitä.
- Peli päättyy, kun pelissä on 50 avutta jäänyttä alusta.

Osaamistavoitteet

Työssä harjoitellaan seuraavia asioita käytännössä:

- Olemassa olevan ohjelmakoodin toimintaan tutustuminen ja sen käyttö tarjottujen rajapintojen avulla (sopimussuunnittelun hyödyntäminen)
- Toiminnallisuuden lisääminen sovellukseen siihen määriteltyjen rajapintojen mukaan (sopimussuunnittelun noudattaminen)
- Graafisen käyttöliittymän toteuttaminen osaksi ohjelmaa
- Poikkeusten käsittely ja virhetilanteiden hallinta
- Oman koodin testaus yksikkötasolla

2. Vaatimukset

Harjoitustyöhön on toteutettava vähintään tietyt perusominaisuudet, jotta sen suoritus voidaan hyväksyä. Lisäksi tarjolla on joukko vapaaehtoisia lisäominaisuuksia, joita toteuttamalla voi korottaa työn arvosanaa.

Pakolliset ominaisuudet

Harjoitustyössä on toteutettava ainakin alla olevat ominaisuudet. Minimitoteutuksen täyttävästä työstä voi ansaita **hyväksytyn** suorituksen (arvosana **1**). Perustoteutukselle määriteltyjen ominaisuuksien mukaisella toteutuksella voi työstä ansaita **korkeintaan arvosanan 3**. Korkeampi arvosana edellyttää myös joidenkin lisäosien toteutusta.

Minimi:

- Ohjelmaan on toteutettu pääkäyttöliittymä, jossa esitetään ainakin pelissä olevat tähtijärjestelmät ja alukset.
- Pelaajan aluksen paikkaa voi vaihtaa välittömästi ilman liikeanimaatiota (esim. kursorin osoittamaan paikkaan tai valittuun tähtijärjestelmään)
- IGalaxystä periytetty Galaxy-luokka läpäisee sille tarjotut yksikkötestit.
- Peli etenee vuoropohjaisesti. Pelin tapahtumien perusteella pelaajalle lasketaan pisteet, ja mahdollisesti peli katsotaan voitetuksi tai hävityksi. Pelille on siis laadittu jonkinlaiset tavoitteet, joihin pelaaja voi pyrkiä.
- Itse toteutettavalle tilastointiluokalle on kirjoitettu yksikkötestit. Luokalta vaadittu toiminnallisuus on määritelty `iStatistics`-rajapinnan avulla.

Perus:

- Peli selviää hallitusti asetustiedostojen käsittelyssä tapahtuvista virheistä.
- Pelin toteutus sisältää ainakin kaksi erillistä ikkunaa ja/tai dialogia.
- Pelin tilan seuranta. Pelin aikana kerättäviä tilastotietoja esitetään pelaajalle siten, että ne päivittyvät pelin edetessä.
- Tähtijärjestelmätiedot. Kartalla tai muuten pelaajalle luettavassa muodossa esitetään tähtijärjestelmistä keskeistä tietoa: esim. populaation koko, pääasiallinen elinkeino, jne.
- Pelaajan alus toimii vuorovaikutuksessa ei-pelattavien alusten kanssa. Ei-pelattavat alukset reagoivat pelaajan aluksen kanssa tapahtuvaan vuorovaikutukseen.

Lisäominaisuudet

Alla olevia lisäominaisuuksia ei tarvitse toteuttaa, mutta hyvästä lisäosan toteutuksesta voi ansaita korotuksen arvosanaan. Lähtökohtaisesti kukin lisäosa on arvoltaan 0,5 arvosanaa. Yksittäisen lisäosan arvoa voidaan kuitenkin tarvittaessa kasvattaa myöhemmin, jos se osoittautuu erityisen työlääksi. Korotuksen saaminen edellyttää, että lisäosa on **dokumentoitu** palautuksessa. Töiden tarkastajat eivät etsi lisäosia palautetusta koodista!

- Peli toimii reaaliaikaisesti.
- Tasainen ruudunpäivitys. Pelialueelle päivitetään tasaisin väliajoin kertyneet muutokset, sen sijaan että jokainen muutos aiheuttaisi erillisen päivityksen. QTimer-luokka voi olla tässä hyödyllinen.
- Minimaalinen ruudunpäivitys. Toimijan tilan muutoksen seurauksena vain sen välitön ympäristö päivitetään pelialueelle, sen sijaan että koko kartta piirrettäisiin uudelleen.
- Vieritettävä kartta. Pelialue ei ole sidottu rajattuun näkymään, vaan

esitettävä alue vaihtuu pelaajan aluksen liikkeiden perusteella.
- Pelaajan aluksen tasainen liike. Välittömän paikanvaihdon sijaan alus käskytetään liikkumaan johonkin suuntaan, minkä jälkeen se etenee sinne sopivalla nopeudella.
- Top10-lista. Peliin toteutetaan Top10-lista, jonka sisältö säilyy pelin sulkemisen ja uudelleenkäynnistyksen yli.
- Paikallinen moninpeli. Pelissä voi olla samanaikaisesti kaksi pelattavaa alusta, joita eri pelaajat voivat hallita, esimerkiksi yksi hiirellä ja toinen näppäimistöllä. Pisteytys voidaan laskea yhdessä tai erikseen.
- Itse toteutettu moottorityyppi ja sitä käyttävä alus.
- Itse toteutetut yksikkötestit on viety osaksi CI-ympäristöä ja ne ajetaan automaattisesti osana CI:tä.
- Oma lisäominaisuus. Ohjelmaan on toteutettu jokin itse keksitty tämän listan ulkopuolinen lisäominaisuus. Omat lisäominaisuudet on hyväksytettävä hyvissä ajoin etukäteen kurssihenkilökunnalla, jotta niistä voisi saada pisteitä. Hyväksyttäminen tehdään Moodlen harjoitustyökeskustelualueella.

3. Toteutus

Työympäristö haetaan Gitlabista. Valmiit toteutuksen osat tulevat sinne saataville.

Ympäristö

Työtä varten toteutettu valmis ohjelmarunko on saatavilla Gitlabista ryhmän omasta tietovarastosta. Tietovaraston kloonaamisen tuloksena pitäisi alla oleva hakemistorakenne.

```
Paikallinen työhakemisto
│
├── GalaxyGame
│    │
│    ├── Assets
│    │    ├── shipnames.txt
│    │    ├── starsystems.json
│    │    └── ...
│    ├── Course
│    │    └── ...
│    ├── CourseTests
│    │    ├── Galaxy
│    │    │    ├── Galaxy.pro
│    │    │    └── galaxytest.cc
│    │    └── ...
│    ├── Student
│    │    ├── Student.pro
│    │    ├── main.cc
│    │    └── ...
│    ├── StudentTests
│    │    ├── Statistics
│    │    │    └── ...
│    │    └── ...
│    └── Galaxygame.pro
└── .gitignore
```

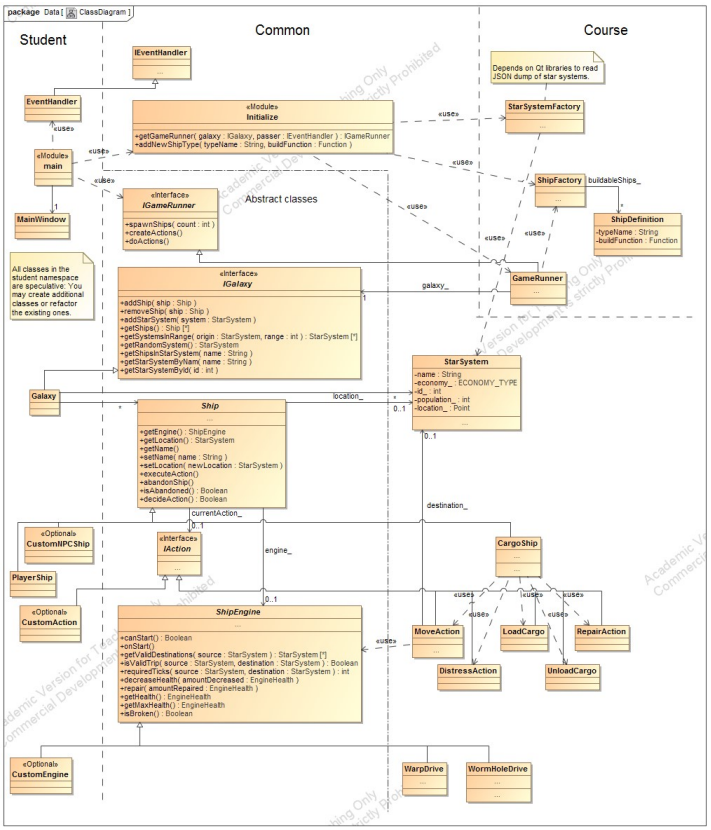
- **GalaxyGame** - Harjoitustyön pääkansio. Sisältää pääprojektitiedoston Galaxygame.pro
- **Assets** - Sisältää esimerkiksi alustukseen liittyviä tiedostoja. Älä muokkaa kurssin tiedostoja. Omia tiedostoja voi lisätä.
- **Course** - Kurssin puolen toteutus. Älä muokkaa tai lisää mitään!
- **CourseTests** - Kurssin puolen testejä. Älä muokkaa tai lisää mitään!
- **Student** - Opiskelijoiden oma toteutus.
- **StudentTests** - Opiskelijoiden omat yksikkötestit. Jokainen yksikkötesti omaan alikansioon.

Valmiiseen toteutuksen osaan (`Course`-aliprojektiin) tulee varmasti jonkin verran muutoksia työn edetessä. Kurssin puolen koodi sekä tarpeelliset rajapintaluokat tullaan jakamaan `template_project`-tietovaraston(https://gitlab.rd.tut.fi/otekn-2017-2018/template_project) kautta. Päivittääksesi kurssin puolen sinun on määritettävä paikalliseen työhakemistoosi kurssin puoli etävarastoksi (remote). Kurssin puolen tietovarasto: `git@gitlab.rd.tut.fi:otekn-2017-2018/template_project.git`

Huomaa, ettei sinulla ole oikeutta tehdä pushia upstreamiin, eikä kurssin puolen koodeja saa muutella tai korjailla itse. Jos huomaat kurssin koodissa jotain bugilta näyttävää, tee siitä tiketti Gitlabin issue trackeriin. Samoin voit tehdä tiketin, jos kaipaat kurssin puolen luokkiin jotain uutta ominaisuutta.

Luokkakaavio

Alla olevassa kuvassa on esitetty ohjelman oleellisimmat luokat.



HUOM! Opiskelijoiden toteuttamien luokkien osalta kaavio on *täysin* kuvitteellinen, luokkia voi olla enemmän (ja on luultavasti syytäkin olla), ja niiden väliset suhteet voivat olla aivan erilaiset.

Sekvenssit

Alle linkatut sekvenssit havainnollistavat, miten luokkakaaviossa esitetyt luokat vuorovaikuttavat. Sekvenssit eivät kata kaikkea, vaan ainoastaan havainnollistavat tärkeimmät osat. **Huom:** MainWindowin ei ole pakko kutsua mitään suoraan, vaan voi myös käyttää jotain muuta oliota välikappaleena.

- Ohjelman tärkeimpien moduulien ja olioiden alustus
- Alusten toimintojen päättäminen ja suorittaminen
- Alusten liikkuminen galaksissa

Valmiina annetut luokat ja rajapinnat

Ohjelmaan toteutetaan kurssin puolesta valmiiksi mm. alla olevat osat:

- Tulossa rajapintojen julkaisun yhteydessä

Itse ohjelmaan on toteutettava ainakin seuraavat osat:

- `main` -pääohjelma, joka avaa pelin ensimmäisen ikkunan ja mahdollisesti alustaa pelin tilan.
- `MainWindow`, ohjelman pääikkuna, jonka kautta pelaajat näkevät pelin tilan ja tekevät siirtonsa.
- Galaksin toteutus `iGalaxy` -rajapinnan mukaisesti.
- `iEventHandler` -rajapinnan toteuttavat tapahtumat
- Jokin muu ikkuna tai dialogi, esim. pelin asetusten säätöön, operaatioiden käynnistämiseen tai pelin lopputuloksen esittämiseen.

Todennäköisesti toteutuksen on syytä sisällyttää näiden lisäksi muitakin luokkia.

Annettujen luokkien ja rajapintojen dokumentaatio

Valmiina annetuille luokille ja rajapinnoille löytyy Doxygen-dokumentaatio.

Testaus

Harjoitustyön hyväksymiseksi vaaditaan, että tilastoinnin toteuttavalle luokalle on kirjoitettu kattavat yksikkötestit iStatistics-rajapinnan perusteella. Laaditut testit palautetaan arvosteltavaksi varsinaisen ohjelmakoodin mukana. Luonnollisesti ohjelman muutkin osat ja ohjelma kokonaisuutena on syytä testata niiden toimivuuden varmistamiseksi, mutta niiden testausta ei erikseen arvostella.

Vaaditut yksikkötestit kirjoitetaan osaksi työympäristöstä löytyvää `TestStatistics` -aliprojektia tiedostoon `tst_teststatistics.cpp`. Mahdollisia muita automoituja testejä varten on luotava omat aliprojektinsa. Näihin on tuotava mukaan tarvittavat tiedostot muista aliprojekteista; katso mallia `TestStatistics.pro` -tiedostosta.

4. Arviointi

Työn arvioinnissa kiinnitetään huomiota seuraaviin tekijöihin:

- Suunnittelu
  - Luokkien vastuualueet ja luokkien suunnittelu
  - Periytymisen käyttö
  - Rajapintamäärittelyt omissa luokissa
- Toteutus
  - Sopimussuunnittelun noudattaminen
  - Poikkeuksien käyttö
  - Muistinhallinta
  - Kapselointi
  - Olio-ohjelmointi ja C++
  - Qt:n käyttö

    - ohj2/opersk-tason asiat
- Toiminnallisuus
    - Toiminnan oikeellisuus
    - Käytön sujuvuus
    - Käyttöliittymän siisteys
- Koodin ulkonäkö & tyyli
    - Vakioiden käyttö
    - Muuttujien alustus
    - Koodin asemointi ja sisennykset
    - Nimeämiskäytännöt
    - Kommentoinnin taso
- Yksikkötestaus
    - Yksikkötestauksen kattavuus (testattavasta luokasta)
    - Yksikkötestien toteutus
- Dokumentaatio
    - Toteutuksen kuvaus
    - Kieliasu
- Versionhallinnan käyttö
    - Committien määrä ja sisältö
    - Lokiviestien sisältö

Viimeksi muutettu: sunnuntai, 22 lokakuu 2017, 12:51

◄ Harjoitustyöryhmien rekisteröinti 28.8.-12.9.

Yleistä tietoa harjoitustyöstä ►

Palaa etusivulle